

CLAIMS

1 1. A computer system that employs a plurality of threads of execution to perform a
2 parallel-execution operation in which the threads identify tasks dynamically and in which
3 the computer system:

- 4 A) provides a global status word that includes a separate status-word field as-
5 sociated with each of the threads; and
6 B) so operates the threads that each thread:
7 i) executes a task-finding routine to find tasks previously identified
8 dynamically and performs tasks thereby found, with the status-
9 word field associated with that thread containing an activity-
10 representing value, until the task-finding routine finds no more
11 tasks;
12 ii) when the task-finding routine finds no more tasks, sets the contents
13 of the status-word field associated with that thread to an inactivity-
14 indicating value;
15 iii) while the status-word field associated with any other thread con-
16 tains an activity-indicating value, searches for a task and, if it finds
17 one, sets the status-word field to the activity-indicating value be-
18 fore attempting to execute a task; and
19 iv) if none of the status-word fields contains an activity-indicating
20 value, terminates its performance of the parallel-execution opera-
21 tion.

1 2. A computer system as defined in claim 1 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 3. A computer system as defined in claim 1 wherein:

- 2 A) each thread has associated with it a respective work queue in which it
3 places task identifiers of tasks that identifies dynamically;

4 B) the task-finding routine executed by an executing thread includes per-
5 forming an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifi-
7 ers that the executing thread can claim, thereafter performing a further
8 search for a task identifier in at least one other task-storage location.

1 4. A computer system as defined in claim 3 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 5. A computer system as defined in claim 3 wherein the at least one other task-
2 storage location includes at least one work queue associated with a thread other than the
3 executing thread.

1 6. A computer system as defined in claim 5 wherein:

- 2 A) there is a size limit associated with each work queue;
- 3 B) when a given thread dynamically identifies a given task that would cause
4 the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were
6 placed in that work queue, the given thread instead places that task identi-
7 fier in an overflow list instead of in that work queue; and
- 8 C) the at least one other task-storage location includes at least one such over-
9 flow list.

1 7. A computer system as defined in claim 5 wherein the task-finding routine in-
2 cludes selecting in a random manner the at least one work queue associated with a thread
3 other than the executing thread.

1 8. A computer system as defined in claim 5 wherein the further search includes re-
2 peatedly searching a work queue associated with a thread other than the executing thread
3 until the executing thread thereby finds a task or has performed a number of repetitions
4 equal to a repetition limit greater than one.

1 9. A computer system as defined in claim 8 wherein the task-finding routine in-
2 cludes selecting in a random manner the at least one work queue associated with a thread
3 other than the executing thread.

1 10. A computer system as defined in claim 3 wherein:

- 2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would cause
4 the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were
6 placed in that work queue, the given thread instead places that task identi-
7 fier in an overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one such over-
9 flow list.

1 11. A computer system as defined in claim 1 wherein the status word fits in a memory
2 location accessible in a single machine instruction.

1 12. A computer system as defined in claim 11 wherein the parallel-execution opera-
2 tion is a garbage-collection operation.

1 13. A computer system as defined in claim 11 wherein each status-word field is a sin-
2 gle-bit field.

1 14. A computer system as defined in claim 13 wherein the activity-indicating value is
2 a logic one and the inactivity-indicating value is a logic zero.

1 15. For employing a plurality of threads of execution to perform a parallel-execution
2 operation in which the threads identify tasks dynamically, a method comprising:

- 3 A) providing a global status word that includes a separate status-word field
4 associated with each of the threads; and

- 5 B) so operating the threads that each thread:
- 6 i) executes a task-finding routine to find tasks previously identified
- 7 dynamically and performs tasks thereby found, with the status-
- 8 word field associated with that thread containing an activity-
- 9 representing value, until the task-finding routine finds no more
- 10 tasks;
- 11 ii) when the task-finding routine finds no more tasks, sets the contents
- 12 of the status-word field associated with that thread to an inactivity-
- 13 indicating value;
- 14 iii) while the status-word field associated with any other thread con-
- 15 tains an activity-indicating value, searches for a task and, if it finds
- 16 one, sets the status-word field to the activity-indicating value be-
- 17 fore attempting to execute a task; and
- 18 iv) if none of the status-word fields contains an activity-indicating
- 19 value, terminates its performance of the parallel-execution opera-
- 20 tion.

1 16. A method as defined in claim 15 wherein the parallel-execution operation is a
2 garbage-collection operation.

1 17. A method as defined in claim 15 wherein:

- 2 A) each thread has associated with it a respective work queue in which it
- 3 places task identifiers of tasks that identifies dynamically;
- 4 B) the task-finding routine executed by an executing thread includes per-
- 5 forming an initial search for a task identifiers in the work queue associated
- 6 with the executing thread and, if that work queue contains no task identifi-
- 7 ers that the executing thread can claim, thereafter performing a further
- 8 search for a task identifier in at least one other task-storage location.

1 18. A method as defined in claim 17 wherein the parallel-execution operation is a
2 garbage-collection operation.

1 19. A method as defined in claim 17 wherein the at least one other task-storage loca-
2 tion includes at least one work queue associated with a thread other than the executing
3 thread.

1 20. A method as defined in claim 19 wherein:

- 2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would cause
4 the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were
6 placed in that work queue, the given thread instead places that task identi-
7 fier in an overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one such over-
9 flow list.

1 21. A method as defined in claim 19 wherein the task-finding routine includes se-
2 lecting in a random manner the at least one work queue associated with a thread other
3 than the executing thread.

1 22. A method as defined in claim 19 wherein the further search includes repeatedly
2 searching a work queue associated with a thread other than the executing thread until the
3 executing thread thereby finds a task or has performed a number of repetitions equal to a
4 repetition limit greater than one.

1 23. A method as defined in claim 22 wherein the task-finding routine includes se-
2 lecting in a random manner the at least one work queue associated with a thread other
3 than the executing thread.

1 24. A method as defined in claim 17 wherein:

- 2 A) there is a size limit associated with each work queue;

- 3 B) when a given thread dynamically identifies a given task that would cause
4 the number of task entries in the work queue associated with the given
5 thread to exceed the size limit if a task identifier that identifies it were
6 placed in that work queue, the given thread instead places that task identi-
7 fier in an overflow list instead of in that work queue; and
8 C) the at least one other task-storage location includes at least one such over-
9 flow list.

1 25. A method as defined in claim 15 wherein the status word fits in a memory loca-
2 tion accessible in a single machine instruction.

1 26. A method as defined in claim 25 wherein the parallel-execution operation is a
2 garbage-collection operation.

1 27. A method as defined in claim 25 wherein each status-word field is a single-bit
2 field.

1 28. A method as defined in claim 27 wherein the activity-indicating value is a logic
2 one and the inactivity-indicating value is a logic zero.

1 29. A storage medium containing instructions readable by a computer system to con-
2 figure the computer system to employ a plurality of threads of execution to perform a
3 parallel-execution operation in which the threads identify tasks dynamically and in which
4 the computer system:

- 5 A) provides a global status word that includes a separate status-word field as-
6 sociated with each of the threads; and
7 B) so operates the threads that each thread:
8 i) executes a task-finding routine to find tasks previously identified
9 dynamically and performs tasks thereby found, with the status-
10 word field associated with that thread containing an activity-

11 representing value, until the task-finding routine finds no more
12 tasks;
13 ii) when the task-finding routine finds no more tasks, sets the contents
14 of the status-word field associated with that thread to an inactivity-
15 indicating value;
16 iii) while the status-word field associated with any other thread con-
17 tains an activity-indicating value, searches for a task and, if it finds
18 one, sets the status-word field to the activity-indicating value be-
19 fore attempting to execute a task; and
20 iv) if none of the status-word fields contains an activity-indicating
21 value, terminates its performance of the parallel-execution opera-
22 tion.

1 30. A storage medium as defined in claim 29 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 31. A storage medium as defined in claim 29 wherein:
2 A) each thread has associated with it a respective work queue in which it
3 places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes per-
5 forming an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifi-
7 ers that the executing thread can claim, thereafter performing a further
8 search for a task identifier in at least one other task-storage location.

1 32. A storage medium as defined in claim 31 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 33. A storage medium as defined in claim 31 wherein the at least one other task-
2 storage location includes at least one work queue associated with a thread other than the
3 executing thread.

- 1 34. A storage medium as defined in claim 33 wherein:
- 2 A) there is a size limit associated with each work queue;
- 3 B) when a given thread dynamically identifies a given task that would cause
- 4 the number of task entries in the work queue associated with the given
- 5 thread to exceed the size limit if a task identifier that identifies it were
- 6 placed in that work queue, the given thread instead places that task identi-
- 7 fier in an overflow list instead of in that work queue; and
- 8 C) the at least one other task-storage location includes at least one such over-
- 9 flow list.

- 1 35. A storage medium as defined in claim 33 wherein the task-finding routine in-
- 2 cludes selecting in a random manner the at least one work queue associated with a thread
- 3 other than the executing thread.

- 1 36. A storage medium as defined in claim 33 wherein the further search includes re-
- 2 peatedly searching a work queue associated with a thread other than the executing thread
- 3 until the executing thread thereby finds a task or has performed a number of repetitions
- 4 equal to a repetition limit greater than one.

- 1 37. A storage medium as defined in claim 36 wherein the task-finding routine in-
- 2 cludes selecting in a random manner the at least one work queue associated with a thread
- 3 other than the executing thread.

- 1 38. A storage medium as defined in claim 31 wherein:
- 2 A) there is a size limit associated with each work queue;
- 3 B) when a given thread dynamically identifies a given task that would cause
- 4 the number of task entries in the work queue associated with the given
- 5 thread to exceed the size limit if a task identifier that identifies it were
- 6 placed in that work queue, the given thread instead places that task identi-
- 7 fier in an overflow list instead of in that work queue; and

8 C) the at least one other task-storage location includes at least one such over-
9 flow list.

1 39. A storage medium as defined in claim 29 wherein the status word fits in a mem-
2 ory location accessible in a single machine instruction.

1 40. A storage medium as defined in claim 39 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 41. A storage medium as defined in claim 39 wherein each status-word field is a sin-
2 gle-bit field.

1 42. A storage medium as defined in claim 41 wherein the activity-indicating value is
2 a logic one and the inactivity-indicating value is a logic zero.

1 43. A computer signal representing a sequence of instructions that, when executed by
2 a computer system, configures the computer system to employ a plurality of threads of
3 execution to perform a parallel-execution operation in which the threads identify tasks
4 dynamically and in which the computer system:

5 A) provides a global status word that includes a separate status-word field as-
6 sociated with each of the threads; and

7 B) so operates the threads that each thread:

8 i) executes a task-finding routine to find tasks previously identified
9 dynamically and performs tasks thereby found, with the status-
10 word field associated with that thread containing an activity-
11 representing value, until the task-finding routine finds no more
12 tasks;

13 ii) when the task-finding routine finds no more tasks, sets the contents
14 of the status-word field associated with that thread to an inactivity-
15 indicating value;

- 16 iii) while the status-word field associated with any other thread con-
17 tains an activity-indicating value, searches for a task and, if it finds
18 one, sets the status-word field to the activity-indicating value be-
19 fore attempting to execute a task; and
20 iv) if none of the status-word fields contains an activity-indicating
21 value, terminates its performance of the parallel-execution opera-
22 tion.

1 44. A computer signal as defined in claim 43 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 45. A computer signal as defined in claim 43 wherein:

- 2 A) each thread has associated with it a respective work queue in which it
3 places task identifiers of tasks that identifies dynamically;
4 B) the task-finding routine executed by an executing thread includes per-
5 forming an initial search for a task identifiers in the work queue associated
6 with the executing thread and, if that work queue contains no task identifi-
7 ers that the executing thread can claim, thereafter performing a further
8 search for a task identifier in at least one other task-storage location.

1 46. A computer signal as defined in claim 45 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 47. A computer signal as defined in claim 45 wherein the at least one other task-
2 storage location includes at least one work queue associated with a thread other than the
3 executing thread.

1 48. A computer signal as defined in claim 47 wherein:

- 2 A) there is a size limit associated with each work queue;
3 B) when a given thread dynamically identifies a given task that would cause
4 the number of task entries in the work queue associated with the given

thread to exceed the size limit if a task identifier that identifies it were placed in that work queue, the given thread instead places that task identifier in an overflow list instead of in that work queue; and

- C) the at least one other task-storage location includes at least one such overflow list.

49. A computer signal as defined in claim 47 wherein the task-finding routine includes selecting in a random manner the at least one work queue associated with a thread other than the executing thread.

50. A computer signal as defined in claim 47 wherein the further search includes repeatedly searching a work queue associated with a thread other than the executing thread until the executing thread thereby finds a task or has performed a number of repetitions equal to a repetition limit greater than one.

51. A computer signal as defined in claim 50 wherein the task-finding routine includes selecting in a random manner the at least one work queue associated with a thread other than the executing thread.

52. A computer signal as defined in claim 45 wherein:

- A) there is a size limit associated with each work queue;
- B) when a given thread dynamically identifies a given task that would cause the number of task entries in the work queue associated with the given thread to exceed the size limit if a task identifier that identifies it were placed in that work queue, the given thread instead places that task identifier in an overflow list instead of in that work queue; and
- C) the at least one other task-storage location includes at least one such overflow list.

53. A computer signal as defined in claim 43 wherein the status word fits in a memory location accessible in a single machine instruction.

1 54. A computer signal as defined in claim 53 wherein the parallel-execution operation
2 is a garbage-collection operation.

1 55. A computer signal as defined in claim 53 wherein each status-word field is a sin-
2 gle-bit field.

1 56. A computer signal as defined in claim 55 wherein the activity-indicating value is
2 a logic one and the inactivity-indicating value is a logic zero.

1 57. A computer system that employs a plurality of threads of execution to perform a
2 parallel-execution operation in which the threads identify tasks dynamically, the com-
3 puter system including:

- 4 A) means for providing a global status word that includes a separate status-
5 word field associated with each of the threads; and
- 6 B) means for so operating the threads that each thread:
 - 7 i) executes a task-finding routine to find tasks previously identified
8 dynamically and performs tasks thereby found, with the status-
9 word field associated with that thread containing an activity-
10 representing value, until the task-finding routine finds no more
11 tasks;
 - 12 ii) when the task-finding routine finds no more tasks, sets the contents
13 of the status-word field associated with that thread to an inactivity-
14 indicating value;
 - 15 iii) while the status-word field associated with any other thread con-
16 tains an activity-indicating value, searches for a task and, if it finds
17 one, sets the status-word field to the activity-indicating value be-
18 fore attempting to execute a task; and
 - 19 iv) if none of the status-word fields contains an activity-indicating
20 value, terminates its performance of the parallel-execution opera-
21 tion.